Wax

MEMO TO: Packet Radio Group

FROM: Jerry Burchfiel

DATE: 24 MAY 1974

A meeting was held May 1 at BBN to generate a preliminary design specification for a Packet Radio Station, and to discuss issues related to the Packet Radio Terminal. Participants were Dale Walls of Collins Radio, Dave Brandin of Stanford Research Institute, and Jerry Burchfiel and Ray Tomlinson of Bolt Beranek and Newman.

Results of that meeting are as follows:

1. A design specification for the Packet Radio Station. This should be considered preliminary rather than final because Collins is still investigating the feasibility of the packaging and standardization recommendations.

2. A specification for a "Standard Packet Interface" which will be used to interconnect the Repeater processor and the Repeater Transceiver, to interconnect the Repeater Processor and the Station processor, and perhaps to interconnect a host and a high-speed IMP.

These two specifications are enclosed, and comments are welcome.

Packet Radio Temporary Note #104

NIC #30770

Specification of a Standard Packet Interface

May 1, 1974

Dave Brandin, Stanford Research Institute

Jerry Burchfiel, Bolt Beranek and Newman

Ray Tomlinson, Bolt Beranek and Newman

Dale Walls, Collins Radio, Inc.

I. Introduction

There are increasing numbers of applications within the ARPA community where a high-speed interface is required: Packet Radio Station, High Speed IMP host interface, multidrop terminal controller, etc. The choice of a standard interface for all these applications simplifies protocol definitions, simplifies acquisition of the interfaces, and makes it easy to do experiments with these interfaces self-looped or cross-connected.

In all these applications, information moves in discrete packets or messages which must be explicitly delimited by interface signals. In addition, information flow is bidirectional. To support bidirectional information flow, a full duplex interface is specified, rather than a half duplex interface, for two reasons:

(1) A full duplex interface avoids the lockup situations possible with a half duplex interface. (See BBN Report 1822)

(2) A full duplex interface can be self-looped for diagnostic testing and experiments.

Further, the full duplex interface specified is completely symmetric, so self-looping of an interface or cross-coupling of two interfaces requires no logic, only jumper cables.

Digitized by the Internet Archive
in 2025 with funding from
Amateur Radio Digital Communications, Grant 151

https://archive.org/details/temporary-note-104

To support the bandwidth required in these applications (one megabit/sec and higher), the interface is 16 bits parallel in both directions. In order to accommodate processors on either side which have varying data rates, the interface is asynchronous. The interface uses a 4-way (speed independent) handshake in both directions; it is quite similar to the host-IMP interface specified in BBN Report 1822, except that it operates in a word-serial fashion instead of a bit serial fashion.

II. Signal Specification

Assume that the interface is used to interconnect two pieces of equipment, X and Y. The 16 data signals from X to Y are named XDAT15 (most significant bit) to XDAT00 (Least significant bit). Similarly, the signals from Y to X are named YDAT15 to YDAT00. To use the interface in self-loop experiments (eliminating device Y, for instance) jumper all XDATnn outputs back into YDATnn inputs.

Three signals are used to control the asynchronous word transfer from Y to X; READY FOR Y WORD, THERE'S YOUR Y WORD, and Y PACKET DONE. The four-way handshake process proceeds as follows:

(1) When X is ready to accept a word from Y, it asserts READY FOR Y WORD.

(2) When Y has received X's READY FOR Y WORD and has stabilized an output data word on YDAT15-YDAT0, it asserts THERE'S YOUR Y WORD.

(3) When X has received THERE'S YOUR Y WORD and accepted the data on YDAT15-YDAT0, it drops READY FOR Y WORD.

(4) When Y sees the drop in READY FOR Y WORD, it drops THERE'S YOUR Y WORD.

The cycle can begin again when X has seen THERE'S YOUR Y WORD go false, and X is ready to accept another word from Y.

The control signals for transfer in the opposite direction function in the same way; they are READY FOR X WORD, THERE'S YOUR X WORD, and X PACKET DONE. To operate the interface in self-loop mode, (for example, eliminating device Y) jumper READY FOR Y WORD output back into READY FOR X WORD input, and THERE'S YOUR X WORD output back into THERE'S YOUR Y WORD input.

The two PACKET DONE signals delimit the end of a transmission. Again consider a transfer from Y to X. The transfer proceeds as described above until all words have been successfully transferred. X then requests another word by asserting READY FOR Y WORD. Now Y has no more words to supply, so instead of asserting THERE'S YOUR Y WORD, it asserts Y PACKET DONE. The 4-way handshake again proceeds to completion; X will drop READY FOR Y WORD, which causes Y to drop Y PACKET DONE. In the above sequence, THERE'S YOUR Y WORD and Y PACKET DONE behave similarly; these signals may be OR'd together in the handshake logic.

To operate the interface in self-loop mode (for example, eliminating device Y), jumper the X PACKET DONE output back into the Y PACKET DONE input.

There are two other signals in this interface: X RESET and Y RESET. X RESET is asserted by the X device when it sets itself to its ground state; an example of this is an X processor which is being loaded with its interface control program. When X RESET is asserted, communication through the interface will be inhibited by logic in the X device which forces both READY FOR Y WORD and THERE'S YOUR Y WORD to be false. When X RESET finally drops, the state of X's interface driver program is likely to be lost; the protocol which communicates across this interface should then go through its resynchronization sequence. (In the 1822 interface, this involves transmission of two host-IMP NOP messages across the interface in both directions.) The protocol proposed ~~for~~ this interface is

described in Section IV.

Table 1 gives a summary of the 48 interface signals:

	X Source	Y Source
Supervisory:	X RESET	Y RESET
Data:	XDAT00-XDAT15	YDAT00-YDAT15
X to Y control:	READY FOR Y WORD	THERE'S YOUR Y WORD
		Y PACKET DONE
X to Y control:	THERE'S YOUR X WORD	READY FOR X WORD
	X PACKET DONE	

Table 1: Standard Packet Interface Signals

III. Electrical Specification

A suggested electrical standard was also specified; each signal source is provided by a 7440 buffer gate (or equivalent) through a 75 ohm series resistor. Each signal is received in a 74 series gate input, with clamp diodes to prevent excursions below 0 volts or above +5 volts. Each signal is transmitted through a twisted-pair with one side grounded at both ends of the run. This 40 twisted pair interconnection cable can be up to 100 feet in length. Signal conventions are +3 volts: presence of bit, assertion of signal, Ground: absence of bit, negation of signal.

IV. Implementation

The implementation required for most of the signals above is perfectly straightforward. However, this is not the case for X RESET and Y RESET, because the interpretation placed on these signals is rather complex. These two signals have been provided to reliably resynchronize and resume transmission after a temporary lapse of service and possible loss of state information by either system.

The implementation recommended below attempts to make interface programming as simple as possible. This will minimize bugs, and make it possible to create small ROM-bootstrap loaders which load programs through this interface.

First, we will specify the behavior of the X interface; by symmetry, the Y interface will be the same. When the X interface observes Y RESET, the Y system has suffered a possible loss of state, so the message in transit from Y to X is likely to be incomplete. Similarly, the message in transit from X to Y is likely to be incomplete.

Both X and Y must recognize this explicitly by sending a NOP message (defined in the lowest level protocol using this interface) which may be appended to the current incomplete message in both directions. Next, X and Y must both throw away the message currently being received when it completes. This message may be either a NOP or a NOP appended to an incomplete message.

The simplest arrangement for the interface driver is a pair of processes, one sending messages and the other receiving messages. Y RESET must be provided as an error status bit to each process. However, the two processes will need to clear this condition independently; the simplest implementation is an Input Error flop and an Output Error flop. Both flops are set by Y RESET and they are cleared independently under program control.

To insure that messages are not flowing across the interface while Y is being reset, the Y RESET signal inhibits READY FOR X WORD and THERE'S YOUR Y WORD inside the Y interface.

When X is reset, Y behaves in a fashion symmetric to that outlined above. Of course, this X RESET indicates that the state of X's interface driver, as well as the current incoming and outgoing messages, are likely to be lost. The appropriate action is triggered by setting the Error flops in the X interface to cause resynchronization of the message flow in both directions.

The specification for the X interface is thus:

1. It detects Y RESET as a program-readable status condition, but not an interrupt condition.
2. It contains two flops, Input Error and Output Error. They are both set by X RESET or Y RESET. Input Error is cleared by initiation of input, and Output Error is cleared by initiation of output. Both flops are program-readable status conditions, but not interrupt conditions.
3. The signals THERE'S YOUR X WORD and READY FOR Y WORD are inhibited by X RESET.

The X input processes is:

```
ILP:  Wait until Input Buffer is available
      Wait until Y RESET is false
      Start input  //  Clears Input Error
      Wait for input completion
      UNLESS Input Error
      DO Put message on Input Queue
      goto ILP
```

And the X output process is:

```
OLP:  Wait until Output message is available
      Wait until Y RESET is false
      IF Output Error
      THEN (setup NOP for transmission; NOPFlag:=TRUE)
      ELSE setup message for transmission
      Start output  //  Clears Output Error
      Wait for output completion
      IF Output Error
      THEN goto OLP
      UNLESS NOPFlag
      DO Deque message from Output Queue
      NOPFlag :=FALSE
      goto OLP
```